Multi-criteria Search Algorithm: An Efficient Approximate K-NN Algorithm for Image Retrieval

Mehdi Badr, Dan Vodislav, David Picard, Philippe-Henri Gosselin, Shaoyi Yin

▶ **To cite this version:**

Mehdi Badr, Dan Vodislav, David Picard, Philippe-Henri Gosselin, Shaoyi Yin. Multi-criteria Search Algorithm: An Efficient Approximate K-NN Algorithm for Image Retrieval. 2013 IEEE International Conference on Image Processing (ICIP 2013), Sep 2013, Melbourne, Australia. pp.2901-2905, 10.1109/ICIP.2013.6738597 . hal-00832196v2

HAL Id: hal-00832196
https://hal.science/hal-00832196v2

Submitted on 12 Jun 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

hal-00832196, version 1 - 10 Jun 2013

MULTI-CRITERIA SEARCH ALGORITHM: AN EFFICIENT APPROXIMATE K-NN ALGORITHM FOR IMAGE RETRIEVAL

Mehdi Badr, Dan Vodislav, David Picard, Shaoyi Yin

ETIS
University of Cergy-Pontoise, ENSEA, CNRS, France

Philippe-Henri Gosselin

TexMex team
INRIA Rennes, France

ABSTRACT

We propose a new method for approximate k-NN search in large scale image databases, based on top-k multi-criteria search techniques. The method defines a simple index structure based on sorted lists, which provides a good compromise between fast retrieval, storage requirements and update cost. The search algorithm delivers approximate results with guarantees about false negatives, with fast emergence of good approximations, monotonically improved and leading if necessary to an exact result. Experiments with the on-disk implementation show that our method produces very good approximate results several times faster than the Baseline method.

Index Terms— CBIR, k-NN search, image databases, multi-criteria search, top-k algorithms

1. INTRODUCTION

Large Scale Content Based Image Retrieval (LSCBIR) aims at finding in a huge collection of images, the subset of images that are the most visually similar to a given query image.

The most efficient systems rely on the extraction of local features (typically SIFT descriptors). Then, the descriptors of the query are matched with those of the images in the collection, and images with the highest number of matches are considered the most similar (like in [1]). Alternatively, the descriptors are combined into a single vector signature, and the signature of the query is compared to that of the images in the collections (see [2, 3] for example). In either case, the goal is to perform the computation of image similarities as fast as possible. To achieve this nearest neighbor search, indexing structures and the associated search algorithms have been proposed, such as LSH [4], inverted files [2], NV-Tree[5], etc. The main challenges regarding these structures are threefold. First, the query time should be as low as possible. Second, the storage cost of the indexing structure should also be as low as possible. Finally, the update procedure of the index structure (when new images enter the collection) should be as light as possible.

In this paper, we propose MSA (Multi-criteria Search Algorithm), a new indexing and search method for LSCBIR based on top-k multi-criteria search algorithms [6, 7]. These algorithms consider multiple evaluation criteria returning a score for each object in a given collection. The global score is obtained by composing individual scores through an (usually monotone) aggregation function. Top-k multi-criteria algorithms provide efficient methods to find objects with the k best global scores that avoid exhaustive computation.

Our method exploits the multi-dimensional nature of image signatures to reformulate the LSCBIR k-NN search issue as a top-k multi-criteria problem. This contribution provides a good compromise between fast retrieval, storage requirements and update cost. These properties come from a very simple index structure based on sorted lists, easy to create, to update and to distribute. The related search algorithm proposes a new approximate k-NN search method based on top-k multi-criteria techniques, with guarantees about false negatives, with fast emergence of good approximations, monotonically improved and leading if necessary to an exact result.

The proposed method is validated through experimental evaluation over a 1 million images dataset. Implemented with on-disk structures, it provides high-quality approximate results several times faster than the Baseline exact method.

The rest of the paper is organized as follows: after we present the related work in section 2, the section 3 describes our contribution. Then, section 4 reports experimental evaluation of the proposed method before we conclude.

2. RELATED WORK

To perform the retrieval of visually similar images in a collection, highly discriminative local features (or descriptors) are extracted from the images. Then, each feature of a given query is searched for its k nearest neighbors (k-NN) in the set of all features from all images. Each of such neighbors casts a vote for the image it belongs to and the retrieved images are those with the highest number of votes [1]. The cost of an exhaustive k-NN search in high dimensional spaces with an extremely large number of elements (more than a billion) is clearly prohibitive.

Approximate k-NN search methods based on specific index structures have been proposed to overcome this problem. The Locality-Sensitive Hashing [4] is one of the most popular, with the idea to group nearby points into a same hash bucket with high probability thanks to well chosen hash functions. Several extensions of LSH have been proposed, mainly with more appropriate hash functions. Other methods have been proposed with tree structures such as KD-tree [8], NV-tree [5], or inverted files [2].

However, voting based methods need to perform a k-NN search for each descriptor of the query, which is problematic when a large number of descriptors per image is considered. Moreover, all descriptors are often stored in addition to the index structures, which can be prohibitive in storage cost. To address this problem, aggregation strategies that map the set of descriptors of an image to a single vector (called *signature*) have been proposed [9, 3]. The retrieval is then reduced to a single k-NN search for the query image by comparing the signatures. Having a single compact vector per image has clearly a computational and storage advantage. Recently, the authors of [9] proposed a Quantization/Inverted File approach to index such signatures. In this paper, we are interested in top-k multi-criteria algorithms and their associated index structures.

Top-k multi-criteria algorithms have been extensively studied,

hal-00832196, version 1 - 10 Jun 2013

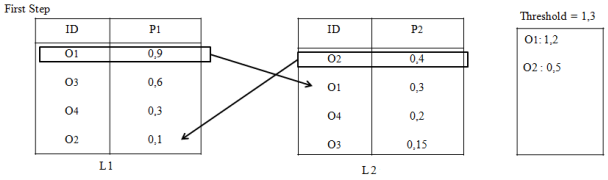

Fig. 1. The Threshold Algorithm strategy

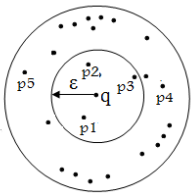

Fig. 2. ϵ-exclusive k-NN search example

e.g. in the context of data integration. Queries are expressed through a set of ranked predicates, each one being independently evaluated and returning a relevance score for any input object. A monotone aggregation function combines partial scores from each predicate into the final object score and objects with the best k scores are returned. Many algorithms have been proposed in this context, considering different types of access to the partial score lists and cost models. The survey [6] presents a rich overview of these techniques.

We consider here algorithms having both sorted and random access to all the score lists, the best known one being TA (Threshold Algorithm) [10]. Figure 1 illustrates the TA strategy over an example of top-k query processing. The example considers two criteria, each one producing a list of scores in the $[0, 1]$ range for the same object collection, with sum as the aggregation score function. TA has for each list both sorted access (in decreasing order of scores) and random access (to a given object). The TA strategy successively considers each list for a sorted access, then for each new discovered object gets the other scores through random access in the other lists. Other strategies [11] use a benefit-based approach and select at each step the "best" list for a sorted access.

TA maintains a list of candidates sorted by decreasing order of the global score and a threshold value representing the highest score that a new object could have from now on. In Figure 1, objects O_1 and O_2 are discovered by sorted access to lists L_1, respectively L_2, then random accesses for the missing scores allow computing their global scores. The threshold value becomes 1.3, because no new object could have scores greater than 0.9 in L_1 and 0.4 in L_2. TA stops when the scores of the best k candidates exceed the threshold value. A new sorted access to L_1 would discover O_3 (global score 0.75), but would also decrease threshold to 1 (0.6+0.4), allowing O_1 (score 1.2) to be safely reported as the best object.

The MSA algorithm is directly inspired by these top-k algorithms, adapted to the MSA index structure.

3. PROPOSED METHOD

The *Multi-criteria Search Algorithm* (MSA) borrows from top-k multi-criteria techniques to perform efficient approximate k-NN search in large-scale image databases. MSA is based on an index structure composed of m ranked lists, one for each dimension of the image signatures. For a given query object, the MSA algorithm exploits this index to generate m independent search processes, merged through a top-k multi-criteria approach.

The general idea of MSA is that the search process for each dimension successively discovers candidates for the global k-NN result. An approximation parameter ϵ controls the moment when the MSA algorithm stops. The returned result is the set of the k best candidates at that moment. As shown below, MSA provides *monotone approximation*; increasing ϵ values lead to a later stop, thus adding new candidates and improving the quality of the approximate result.

We first characterize the approximate k-NN result produced by MSA, then we present the MSA index structure and algorithm.

3.1. ϵ-exclusive Search

We consider a database $\mathcal{D}$ as being a set of points in the normed space l_p^m, representing the Euclidean space R^m with the l_p norm and the distance $d(\cdot, \cdot)$ induced by this norm. For a query object $q \in R^m$ and an approximation parameter $\epsilon > 0$, we note $P_{k,\epsilon}(q) \subset \mathcal{D}$ the approximate k-NN result returned by the MSA algorithm.

Definition 1 *(ϵ-exclusive k-NN search)*
Given a query object q and a threshold $\epsilon > 0$, a k-NN approximate search for q in the database $\mathcal{D}$ is said to be ϵ-exclusive, if, given $P_{k,\epsilon}(q)$ the result of the approximate search and $P_k(q)$ an exact k-NN result, for any $x \in P_k(q) - P_{k,\epsilon}(q)$ we have $d(x, q) \geq \epsilon$.

Intuitively, ϵ-exclusive search guarantees that missed points (false negatives) in the approximate result have "low quality", i.e. they are at distance at least ϵ from q. Note that the ϵ-exclusive condition implies that any $x \in \mathcal{D} - P_{k,\epsilon}(q)$ has $d(x, q) \geq \epsilon$.

Figure 2 illustrates ϵ-exclusive k-NN search. Here k=5 and the approximate result $P_{k,\epsilon}(q) = \{p_1, ..., p_5\}$. Points not returned by the algorithm (including those better than p_4 or p_5) are all at distance at least ϵ from q. Note that points in $P_{k,\epsilon}(q)$ at distance at most ϵ from q (here p_1, p_2, p_3) surely belong to an exact solution $P_k(q)$.

If we note $\zeta = max_{x \in P_{k,\epsilon}(q)}(d(x, q))$ the greatest distance to q for points in the approximate solution, then the quality of the approximate result improves when ϵ grows and ζ decreases. We demonstrate that MSA provides ϵ-exclusive k-NN search with monotone increase of ϵ and decrease of ζ.

3.2. The MSA index structure

Figure 3 presents the index structure used by the MSA algorithm. As mentioned above, we consider a database $\mathcal{D}$ of image signatures (m-dimensional vectors). The MSA index is composed of m sorted lists $L_1, ..., L_m$, one for each dimension. Each element of a list L_i is a couple (x, x_i), where x references an image signature in $\mathcal{D}$ and x_i is the i-th component of that vector. Each list L_i indexes all the elements in $\mathcal{D}$ and is sorted in decreasing order of x_i.

Given a query q=$(q_1, ..., q_m)$, the index enables fast retrieval by binary search of the position corresponding to q_i in the sorted list L_i, for each i. The role of each list L_i is to successively deliver entries (x, x_i) in increasing order of the distance between x_i and q_i, by sequential access from q_i in both directions. At each step, two candidates are considered, up (u_i) and down (d_i) from q_i, at distance Δ_{iu}=$u_i - q_i$, respectively Δ_{id}=$q_i - d_i$ from q_i. The selected candidate is the one closer to q_i, i.e. at distance Δ_i=$min(\Delta_{iu}, \Delta_{id})$. The next entry in the chosen direction will be considered as a candidate for the next step. Note that only one candidate is considered at each step if candidates in the other direction are exhausted.

This corresponds to two access methods provided by the MSA index: (i) $init(q)$ that finds the position of each q_i in L_i, and (ii)

hal-00832196, version 1 - 10 Jun 2013

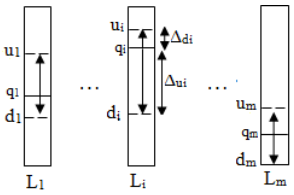

Fig. 3. MSA index structure

$getNext(L_i, q_i)$ that returns the next entry (x, x_i) in L_i in increasing order of distance to q_i.

Although the MSA index may be implemented in main memory, we consider here the case of very large image databases, where the index and the database must be stored on disk. Index lists and the database are both implemented as files with sequential/random access. The database file sequentially stores the signature vectors, the ordinal number in this sequence being used in the index entries as a reference to the signature. Alternatively, index lists may be implemented as B+ trees in order to optimize updates.

We emphasize the good properties of the MSA index structure (i) for *search*, with logarithmic binary search for *init* and constant time for *getNext*, (ii) for *creation* and *update*, as simple as for sorted lists or B+ trees, (iii) for *distribution*, with independent index lists that could be stored on different sites, and (iv) for *storage needs*, which are of $m \times n \times (|ref| + |comp|)$, where n is the number of signatures, m their dimensionality, $|ref|$ the size of an object reference and $|comp|$ the size of a signature component.

3.3. Multi-criteria Search Algorithm

The MSA algorithm, presented in Algorithm 1, uses a top-k multi-criteria approach to merge information about the most promising candidates returned by the individual index lists. Inputs are the query vector q, the ϵ approximation parameter and the number of results k. MSA maintains a list of candidates $cand$ sorted in ascending order of the distance to q and a threshold ϵ_{crt}.

The monotone generic aggregation function $\mathcal{F}$ computes the distance between two vectors by combining distances on each dimension. It may be instantiated e.g. to the l_p distance $d(x, y) = \sqrt[p]{\sum_{i=1}^{m} |x_i - y_i|^p}$. Threshold ϵ_{crt} represents the minimum distance to q that a new candidate could have; at each moment it is computed by merging through $\mathcal{F}$ the current Δ_i for each dimension. Also, the initial search position in the index lists is fixed through a call to the *init* method.

At each iteration, MSA first selects a dimension i to be probed. Different strategies may be implemented through the *ChooseDimension* function, e.g. the TA strategy that successively probes all the dimensions, or some benefit-based strategy selecting the "best" dimension at that moment. Then, the index list L_i is probed by *getNext* to get the next best candidate. If L_i has no more entries ($x = nil$), the loop stops because all the database objects have been considered. If candidate x returned by L_i is a new one, it is added to the $cand$ list after its distance to q is evaluated by the *ComputeDistance* function, which accesses the database to get the signature of x. Finally, threshold ϵ_{crt} is updated after the new value of Δ_i is computed.

The algorithm stops when there are at least k candidates and, either threshold ϵ_{crt} exceeds the approximation parameter ϵ, or ϵ_{crt} exceeds ζ_{crt} = *KthDistance*($cand$), the k-th best distance. The latter case corresponds to an exact k-NN result, since any new candidate would be at distance at least ϵ_{crt} from q, i.e. it could not improve the k-th best distance in $cand$. In all the cases, the result returned by MSA is composed of the k best candidates *TopK*($cand$).

Algorithm 1 MSA algorithm

Require: $q \in R^m$ and $\epsilon \in R_+^*$ and $k \in N^*$
  $cand \leftarrow \emptyset$
  $\Delta_i \leftarrow 0, i = \overline{1, m}$
  $\epsilon_{crt} \leftarrow \mathcal{F}(\Delta_1, ..., \Delta_m)$
  init(q)
  repeat
    $i \leftarrow$ *ChooseDimension*()
    $(x, x_i) \leftarrow$ *getNext*(L_i, q_i)
    exit loop when $x = nil$
    if x not in $cand$ **then**
      $d \leftarrow$ *ComputeDistance*(x, q)
      AddCandidate($cand, x, d$)
    end if
    $\Delta_i \leftarrow |x_i - q_i|$
    $\epsilon_{crt} \leftarrow \mathcal{F}(\Delta_1, ..., \Delta_m)$
  until $|cand| \geq k$ **and** ($\epsilon_{crt} \geq \epsilon$ **or** $\epsilon_{crt} \geq$ *KthDistance*($cand$))
  return *TopK*($cand$)

The monotonicity of the aggregation function $\mathcal{F}$ and the property of the MSA index to deliver candidates with increasing values of Δ_i guarantee that threshold ϵ_{crt} always increases and that unseen signatures are at distance at least ϵ_{crt} from q. Given these remarks, the following properties are true for the MSA algorithm:

Property 1 *The MSA algorithm provides ϵ-exclusive k-NN search.*

Indeed, any false negative x being an unseen object, $d(x, q) \geq \epsilon_{crt}$, but $\epsilon_{crt} \geq \epsilon$ when MSA stops, thus, $d(x, q) \geq \epsilon$.

Property 2 *MSA is monotonic in ϵ and in time: (i) the execution of MSA(q, ϵ_1, k) corresponds to an early stopping of MSA(q, ϵ_2, k) if $\epsilon_1 < \epsilon_2$; (ii) increasing execution durations correspond to increasing ϵ values and to decreasing ζ values.*

Point (i) comes from the monotonic increase of ϵ_{crt} in time and from the fact that *TopK*($cand$) at some moment corresponds to the result of MSA(q, ϵ_{crt}, k). Point (ii) is true because ϵ_{crt} increases in time, while ζ_{crt} decreases because new candidates added to $cand$ improve the k-th best distance. Note that monotonicity enables an alternative approach in practice, by fixing a maximum execution time if the value of ϵ is difficult to decide.

4. EXPERIMENTS

We use the Holidays database [9] to report the experimental evaluation of our method. The Holidays dataset contains images drawn from personal holidays photos, created to test methods of near-duplicate search. It contains 1491 images gathered in 500 subgroups, each group representing a distinct scene or object (Figure 4). Images in this database are in high resolution color. The Holidays dataset also includes a set of SIFT descriptors. For our experiments, we used the Holidays database artificially enlarged with a set of 1 million unrelated images from the Flickr-10M database. For each image, we computed a single compact VLAT signature of 64 dimensions, such as in [3]. Note that signature vectors are normalized.

Tests are performed on a computer with 4GB of DDR memory and dual core cpu (2.13GHz), under Linux. The MSA index is implemented on disk as described in section 3, with one file per sorted list and one file for the signature database. Algorithms are

hal-00832196, version 1 - 10 Jun 2013

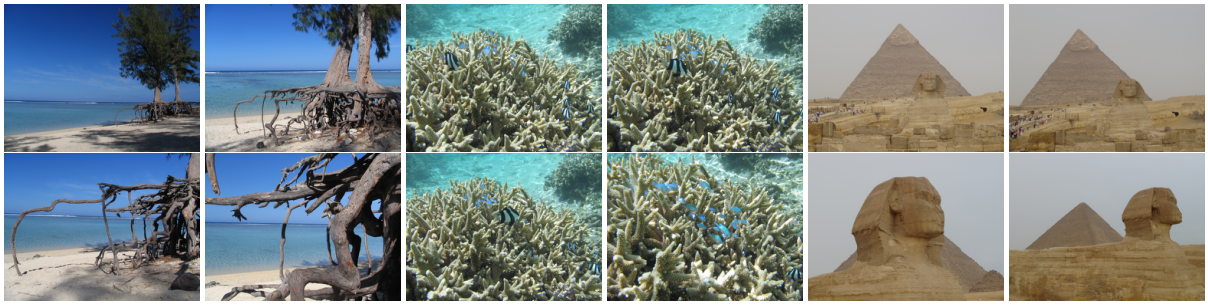

Fig. 4. Images from the Holidays dataset

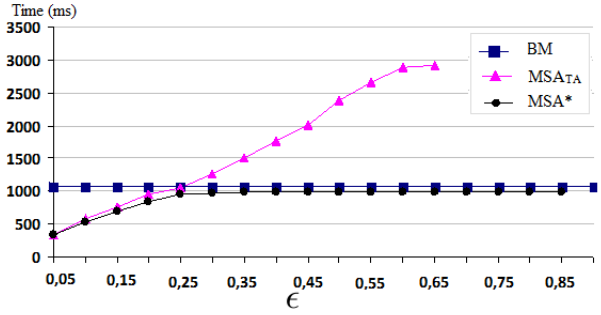

Fig. 5. Execution time for MSA variants as a function of ϵ, compared with Baseline(BM)

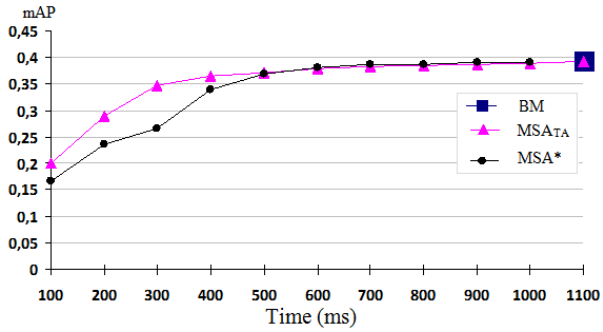

Fig. 6. Search mAP for MSA variants as a function of time, compared with Baseline(BM)

programmed in Java, using the *java.nio* package that already optimizes disk I/O.

Evaluated algorithms

Experiments compare two variants of MSA, called MSA$_{TA}$ and MSA*, with the Baseline method. We report average measures over 500 queries, each query being a signature extracted from each of the Holidays dataset subgroups. The value of k for all the queries is 10. Comparison between images uses the l_2 distance.

MSA$_{TA}$ uses the TA strategy to successively access index lists for getting candidates. MSA* tries to limit I/O by looking for candidates in only one of the index lists. In this case, the *ChooseDimension* function always returns the same value. The selected list is that with the greatest amplitude for the indexed value, as being potentially the most discriminant for the candidates.

The Baseline method (BM) sequentially scans the signature database on disk, computes distance to the query signature and keeps the k best results.

Experimental results

We first analyze the variation of the execution time with ϵ for MSA and compare it with the Baseline method. Figure 5 shows that the execution time grows with ϵ in a different fashion for MSA$_{TA}$ and MSA*. The execution time of the Baseline method (BM) is around 1100 ms, independent of ϵ. For small values of ϵ (less than 0.25), MSA$_{TA}$ and MSA* are faster than BM and their variation curves are very close. The difference between them appears for larger ϵ values; while MSA* remains slightly better than BM even for exact results obtained with large ϵ values, MSA$_{TA}$ performance degrades until almost three times worse than BM for exact k-NN. This may be explained by the extra I/O swapping of MSA$_{TA}$, induced by the access to 64 index files, avoided by MSA*, which uses a single list.

In conclusion, MSA clearly outperforms the Baseline method for small values of ϵ, but is less appropriate for exact k-NN. However, MSA has the advantage to monotonically improve in time its approximate result, going if necessary to an exact result in a time slightly better than the Baseline method (for the MSA* variant).

Finally, we measure the quality of the MSA approximate results in the case where MSA is faster than the Baseline method. Figure 6 represents the mean Average Precision of MSA results, as a function of the execution time. The Baseline method appears as a single point, the dark square on the right. We notice that MSA continuously improves the quality of its results and that MSA$_{TA}$ produces a better approximation than MSA* for small ϵ values, because it considers the best candidates from all the dimensions. MSA$_{TA}$ reaches very good mAP values (0.35 vs 0.39 for BM) already after 300 ms for MSA$_{TA}$ and after 450 ms for MSA*, vs 1100 ms for BM.

In conclusion MSA may deliver very good quality approximate results several times faster than the Baseline method (3-4 times faster in our case).

5. CONCLUSION

In this paper we proposed MSA, a new method for approximate k-NN search for large scale CBIR, based on top-k multi-criteria search techniques. MSA uses a simple index structure that provides a good compromise between fast retrieval, storage requirements and update cost. The MSA algorithm delivers ϵ-exclusive approximate k-NN results with guarantees about false negatives, with fast emergence of good approximations, monotonically improved and leading if necessary to an exact result. Experiments with on-disk implementation of MSA over a 1 million image database show that MSA produces very good approximate results several times faster than the Baseline method.

hal-00832196, version 1 - 10 Jun 2013

6. REFERENCES

[1] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool, "A comparison of affine region detectors," *IJCV*, vol. 65, no. 1–2, pp. 43–72, November 2005.

[2] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *ICCV*, 2003, pp. 1470–1477.

[3] R. Negrel, D. Picard, and P. Gosselin, "Compact tensor based image representation for similarity search," in *ICIP*, 2012.

[4] L. Paulevé, H. Jégou, and L. Amsaleg, "Locality sensitive hashing: A comparison of hash function types and querying mechanisms," *Pattern Recognition Letters*, vol. 31, no. 11, pp. 1348–1358, 2010.

[5] H. Lejsek, B. Jonsson, and L. Amsaleg, "Nv-tree: nearest neighbors at the billion scale," in *Proceedings of ICMR*, 2011, pp. 54:1–54:8.

[6] F. Ilyas, G. Beskales, and M. Soliman, "A survey of top-*k* query processing techniques in relational database systems," *ACM Comput. Surv.*, vol. 40, no. 4, 2008.

[7] M. Badr and D. Vodislav, "A general top-k algorithm for web data sources," in *Proceedings of DEXA*, 2011, pp. 379–393.

[8] J. Friedman, J. Bentley, and R. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM TOMS*, vol. 3, no. 3, pp. 209–226, 1977.

[9] H. Jégou, F. Perronnin, M. Douze, C. Schmid, et al., "Aggregating local image descriptors into compact codes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 9, pp. 1704–1716, 2012.

[10] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," in *Proceedings of PODS*. 2001, pp. 102–113, ACM.

[11] U. Güntzer, W. Balke, and W. Kießling, "Optimizing multi-feature queries for image databases," in *VLDB*, 2000, pp. 419–428.